SoftwareX 29 (2025) 101995

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

SmartRPA: Generating software robots from user interface logs

S. Agostinelli [a,*], T. Hohenadl [b], A. Marrella [a], A. Martínez-Rojas [c]

[a] Sapienza University, Rome, Italy
[b] KU Eichstätt-Ingolstadt, Ingolstadt, Germany
[c] Universidad de Sevilla, Sevilla, Spain

ARTICLE INFO

Keywords:
Robotic Process Automation (RPA)
Automated RPA Script Generation
SW Robots
User Interface (UI) Logs

ABSTRACT

Robotic Process Automation (RPA) is a maturing technology in the field of Business Process Management (BPM) that automates intensive routine tasks previously performed by a human user on the User Interface (UI) of a computer system, by means of a software robot. To date, RPA tools available in the market strongly rely on the ability of human experts to manually implement the routines to automate. This work addresses the limitations of current manual RPA development by introducing SmartRPA, a cross-platform software tool. SmartRPA analyzes UI logs of past routine executions to generate software robots capable of handling intermediate user inputs, thereby reducing development time and error rates.

1. Motivation and significance

The traditional workflow to conduct an RPA project (cf. Fig. 1) can be summarized as follows [1,2]: (1) determine which process steps are good candidates to be automated; (2) model the selected routines in the form of *flowchart diagrams*, which involve the specification of the actions, routing constructs, data flow, etc. that define the behavior of a software robot; (3) record the mouse/key events that happen on the UI of the user's computer system; (4) develop each modeled routine by generating the software code required to concretely enact the associated software robot on a target computer system; (5) deploy the software robots in their environment to perform their actions; and (6) monitor the performance of software robots to detect bottlenecks and exceptions.

Although RPA [3] is generally considered to be an easy-to-implement technology, in practice, in-depth knowledge is necessary to create reliable and scalable software robots, in particular when specific user inputs are required to properly progress the execution of a routine [4]. As a result, between 30% and 50% of initial RPA implementations are estimated to fail [5,6]. Consequently, a tool that simplifies the realization of an RPA project, particularly the generation of software robots in the presence of many routine variants, can be considered as a relevant artifact to investigate. This leads to the following research questions: (**RQ1**) Which steps are required to make the generation of software robots less dependent on the intervention of RPA human experts? (**RQ2**) How can the understanding of the process behavior recorded in a UI log be improved in order to facilitate the consequent generation of software robots? (**RQ3**) How can irrelevant user actions be effectively filtered out from UI logs to enhance the automatic generation of software robots?

In answering these questions, we present SmartRPA [7,8,9], an open-source software tool (cf. Table 1) that contributes to three research challenges that were put forward in [10,11,12,13,14], namely: **(C1)** the automated identification of the routine steps to robotize from a UI log, **(C2)** the automated detection of all the routine variants that require some user input to proceed with their execution, and **(C3)** the automated generation of executable RPA scripts for enacting software robots at run-time.

To address **C1**, the works [15,16] introduce an approach, along with an implemented tool, that leverages process mining techniques to *(i)* track UI actions performed within Excel and Google Chrome and log them as events, and *(ii)* identify routine fragments that could potentially be automated by a third-party RPA tool. Similarly, [17] presents the Desktop Activity Mining tool, which records user actions during office-based tasks on a UI and generates a process model representing the routine behavior. It is important to note that the Desktop Activity Mining tool relies on capturing mouse click coordinates and storing them in a dedicated UI log, which limits its portability since it cannot replicate observed user behavior consistently across different computer systems. Conversely, SmartRPA is designed to replicate the behavior across diverse environments and settings, covering a wide range of user actions. Additionally, it offers the capability to tag actions as relevant or

* Corresponding author.
E-mail addresses: agostinelli@diag.uniroma1.it (S. Agostinelli), tom.hohenadl@stud.ku.de (T. Hohenadl), marrella@diag.uniroma1.it (A. Marrella), amrojas@us.es (A. Martínez-Rojas).

https://doi.org/10.1016/j.softx.2024.101995
Received 13 March 2024; Received in revised form 11 November 2024; Accepted 25 November 2024
Available online 19 December 2024

irrelevant streamlining the filtering process of those user actions that do not contribute to the task being performed and integrates screenshots to improve the understanding of routines being executed. By analyzing screenshots of user interactions, analysts can identify potential pathways or human reasoning beyond the recorded event sequence, providing a more complete picture of the routine under analysis [18,19].

To tackle **C2**, the authors in [20] propose a self-learning approach to automatically detect high-level RPA rules from historical low-level behavior logs. This approach uses an if-then-else deduction logic to infer rules from behavior logs by identifying relationships between various routines previously performed. These inferred rules are then utilized to facilitate the instantiation of software robots. A similar methodology is adopted in [21], where the FlashExtract framework is introduced. FlashExtract enables the extraction of relevant data from semi-structured documents based on input-output examples, allowing users to uncover the relationships underpinning routine operations. Finally, in [22], the authors detect repetitive edits in text documents by tracking an edit graph and subsequently suggest automation rules for software robots. SmartRPA contributes to this challenge with an algorithm capable of automatically identifying all routine variants that can be emulated by a software robot. This algorithm is extensively detailed in [9]. Regarding **C3**, the literature offers only one relevant solution, known as Robidium [23], which addresses this challenge. Robidium is both an approach and an open-source tool that generates executable scripts solely by interpreting UI logs, allowing these scripts to be executed by the commercial RPA tool UI Path.[1] A key feature of Robidium is that it automates only the most frequent routine variant found in the UI log. This is because Robidium produces RPA scripts that do not require any intermediate user inputs during execution, in contrast to SmartRPA. By intermediate user input, we mean the input required by the user for navigating the routine enactment in presence of multiple choices.

Furthermore, differently from existing literature, SmartRPA advances the state of the art by tackling all three challenges simultaneously rather than individually.

In the scope of this paper, a UI log in its raw form consists of a timestamped sequence of events recorded during one user session, as depicted in Fig. 2. Such events include all the user actions required to accomplish one routine using the UI of one or many software application/s.

In this context, we define a *routine variant* as a specific execution of a routine that differs from the other executions of the same routine by at least one event. An event refers to the enactment of a user action

Table 1
Code metadata.

Nr.	Code metadata description	Please fill in this column
C1	Current code version	2.0
C2	Permanent link to code repository	https://github.com/bpm-diag/smartRPA
C3	Permanent link to Reproducible Capsule	https://doi.org/10.5281/zenodo.11386487
C4	Legal Code License	MIT License
C5	Code versioning system used	Git
C6	Software code languages, tools, and services used	Javascript, Python 3
C7	Compilation requirements, operating environments & dependencies	requirements.txt available
C8	If available Link to developer documentation/manual	https://bpm-diag.github.io/smartRPA/index.html
C9	Support email for questions	agostinelli@diag.uniroma1.it

(coupled with some execution data, like the name of the application where the action occurred, etc.) within a specific routine execution recorded in a UI log at a specific moment in time. The presence of different events in many routine executions may potentially determine alternative behaviors of the routine itself. This is particularly true when some events are triggered only by intermediate user inputs provided at the time of the routine execution. These events act as a *variation point* of the routine, where a choice needs to be made between multiple possible variants.

SmartRPA takes inspiration from the Robotic Process Mining (RPM) framework [13]. RPM aims to support analysts to produce executable specifications of routines, in the form of software robots, interpreting the routine executions stored in a UI log. Specifically, RPM envisions a pipeline of three main stages that consist of: *(i)* collecting and pre-processing UI logs corresponding to executions of one/more routine (s); *(ii)* identifying and discovering candidate routines to be automated with RPA tools; and *(iii)* generating executable RPA scripts. SmartRPA incorporates the three main stages of the RPM framework within a larger approach that includes five operational steps to be applied in sequence: *(i)* Log Recording, *(ii)* Log Processing, *(iii)* Event Abstraction, *(iv)* Process Discovery, and *(v)* Script Generation. Note that such methodological steps serve as our answer to **RQ1**.

Unlike previous versions, the newer version of SmartRPA is able to: *(i)* correlate each event generated by the Action Logger module with a screenshot of the UI where the event occurred, and *(ii)* capture feedback immediately after each action using an annotation feature to tag actions as necessary or not, thus answering respectively both **RQ2** and **RQ3**.

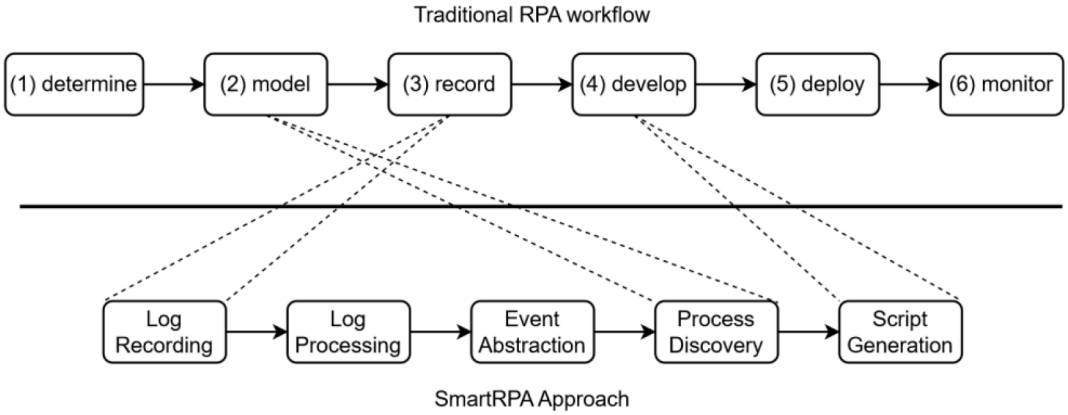


Fig. 1. Traditional RPA Workflow vs. SmartRPA Approach.

In summary, considering the traditional RPA workflow depicted in Fig. 1, SmartRPA enables the analysis of UI logs enriched with screenshots and annotations that keeps track of many routine executions (cf.

[1] www.uipath.com

	A	B	C	D	E	F	G	H	I	J	K	L
1	case:concept:name	case:creator	lifecycle:transition	time:timestamp	org:resource	category	application	concept:name	event_src_path	event_dest_path	clipboard_content	mouse_coord
2	6,23125E+14	SmartRPA by	complete	2024-10-23T12:54:52.993	simone	Browser	Chrome	enableBrowserExtension				
3	6,23125E+14	SmartRPA by	complete	2024-10-23T12:54:52.998	simone	Browser	Chrome	newWindow				
4	6,23125E+14	SmartRPA by	complete	2024-10-23T12:54:52.998	simone	Browser	Chrome	newTab				
5	6,23125E+14	SmartRPA by	complete	2024-10-23T12:54:52.999	simone	Browser	Chrome	startPage				
6	6,23125E+14	SmartRPA by	complete	2024-10-23T12:54:56.352	simone	Browser	Chrome	typed				
7	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:02.760	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	Simone Agostinelli	
8	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:05.797	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	Simone Agostinelli	
9	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:11.807	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	Assistant Professor	
10	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:17.332	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	Assistant Professor	
11	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:21.012	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	agostinelli@diag.uniroma1.it	
12	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:24.181	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	agostinelli@diag.uniroma1.it	
13	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:30.613	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	GSTSMN93M11H501D	
14	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:33.997	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	GSTSMN93M11H501D	
15	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:37.648	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	DIAG Sapienza	
16	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:40.662	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	DIAG Sapienza	
17	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:50.188	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	06:00	
18	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:52.845	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	06:00	
19	6,23125E+14	SmartRPA by	complete	2024-10-23T12:55:55.948	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	23:59	
20	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:04.764	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	23:59	
21	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:10.372	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	Copenaghen (DK)	
22	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:13.255	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	Copenaghen (DK)	
23	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:16.975	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	train+autobus+plane	
24	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:20.068	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	train+autobus+plane	
25	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:23.175	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	Participation at ICPM-24 conference	
26	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:26.293	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	Participation at ICPM-24 conference	
27	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:32.814	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	No	
28	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:36.467	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	No	
29	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:39.778	simone	Clipboard	Clipboard	copy	C:\Users\simon\	Desktop\travel.xlsx	1607 EURO	
30	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:42.713	simone	Browser	Chrome	paste		https://bpm-diag.github.io/form/	1607 EURO	
31	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:45.695	simone	Browser	Chrome	clickRadioButton		https://bpm-diag.github.io/form/		470,306
32	6,23125E+14	SmartRPA by	complete	2024-10-23T12:56:46.762	simone	Browser	Chrome	clickRadioButton		https://bpm-diag.github.io/form/		458,38

Fig. 2. Snapshot of a UI log captured during the execution of the illustrative example.

step "(3) record"), and automatically generate software robots that emulate the most suitable routine variant for any specific intermediate user input required during the routine execution (cf. step "(4) develop"); thus skipping completely the manual modeling activity of the flowchart diagrams (cf. step "(2) model").

2. Software description

2.1. Software architecture

The architecture underlying SmartRPA (cf. Fig. 3) implements the five operational steps discussed in Section 1 as five main software components in Python: *(i)* Log Recording, *(ii)* Log Processing, *(iii)* Event Abstraction, *(iv)* Process Discovery, and *(v)* Script Generation. In Section 2.2 we provide the main functionalities of each component of the architecture.

2.2. Software functionalities

The first software component of the architecture implements the **Log Recording** step and is able to record different types of UI actions from multiple software applications during the enactment of the routine under study. Specifically, a training session in which several users perform the routine to be automated is required to record the UI actions involved in its execution. The Log Recording component provides a Graphical User Interface (GUI) that allows a user to select which software applications s/he wants to record UI actions on (cf. Fig. 4). This software component provides three different types of logging modules: *(i)* a *System Logger,* able to detect those UI actions not related to specific software applications; *(ii)* an *Office Logger,* able to detect the UI actions performed within Microsoft Office applications; and *(iii)* a *Browser Logger,* able to detect the UI actions on web browsers.

The UI actions recorded by the logging modules are sent to a Logging Server, implemented with the *Flask* framework,[2] in charge to store and organize the actions as *events*[3] into several CSV event logs, i.e., the UI logs.

It is worth noticing that multiple users can run the Log Recording component on their computer system many times performing the routine in different training sessions. Each CSV event log contains exactly one long trace of UI actions performed in a single training session by a single user. Technically speaking, *(i)* system events are captured using *PythonCOM* (for Windows APIs and COM objects) and *MacFSEvents* (for MacOS); *(ii)* events generated by Microsoft Office applications are captured using the Office JavaScript APIs; and *(iii)* browser events are captured using JavaScript web extensions developed for each supported web browser.

Each event generated by the logging modules is also correlated with a screenshot of the UI where the event has been generated. Mining screenshot data alongside the aforementioned logging modules can be valuable for generating translucent event logs [24]. By analyzing screenshots of user interactions, analysts can identify potential paths or human reasoning beyond the recorded event sequence, providing a more complete picture of the process [18,19]. To further enhance efficiency, users can capture feedback directly after each action using an annotation feature. This feature allows users to tag actions as necessary or not, streamlining the SmartRPA filtering process and reducing the need for developer supervision to create a better return on investment in the RPA implementation process [25]. Screenshots and supervision features are disabled by default, but they can be activated by clicking on the preferences menu as depicted in Fig. 5.

The second software component of the architecture is the **Log Processing** tool that is triggered when any training session is considered completed. Specifically, after *n* training sessions, the Logging Server will deliver the *n* created CSV event logs to the Log Processing component, in charge of importing them into a single Pandas dataframe.[4] A dataframe is a two-dimensional size-mutable and heterogeneous tabular data structure with labeled axes, which is used as the main artifact to represent event logs in SmartRPA. The dataframe created by the Log Processing component consists of low-level events with fine granularity associated one by one with a recorded UI action, including several columns representing the payload of the recorded event, i.e.: the timestamp, the application that generated the event, the resources involved, which altogether represent the context and data parameters of a user

[2] https://palletsprojects.com/p/flask

[3] For a complete list of logged events, please refer to the following link: https://github.com/bpm-diag/smartRPA/blob/master/src/images/SmartRPA_events.pdf

[4] https://pandas.pydata.org/

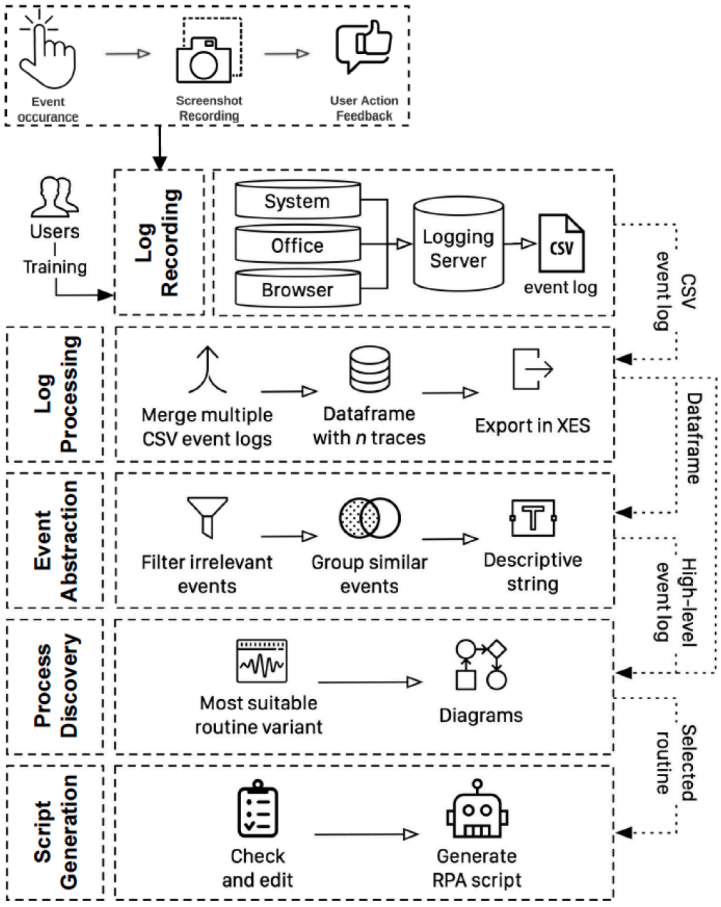
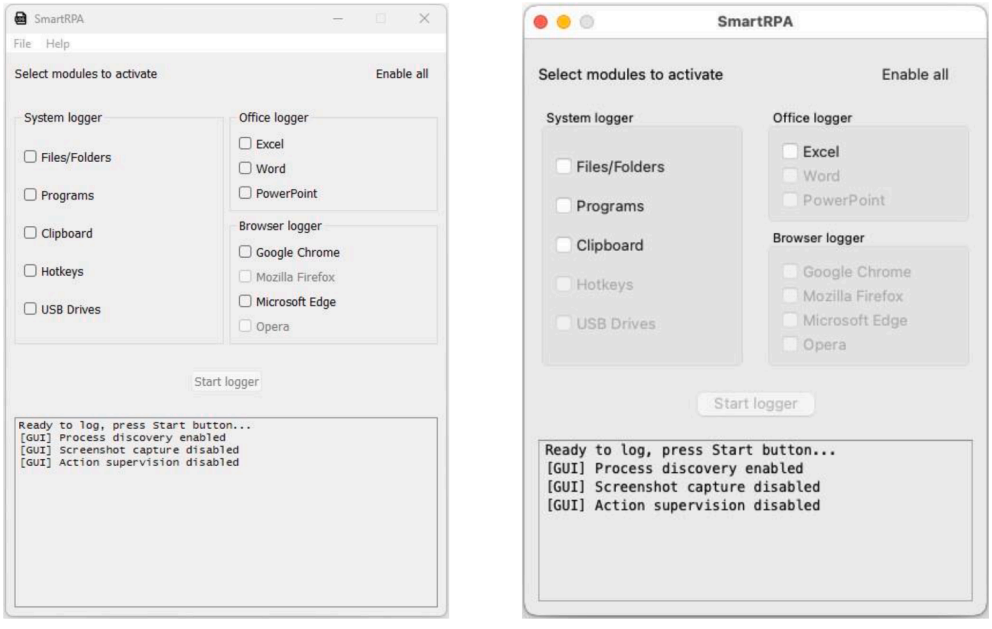


Fig. 3. SmartRPA architecture.



(a) Windows (b) MacOS

Fig. 4. SmartRPA GUI on Windows and MacOS platforms.

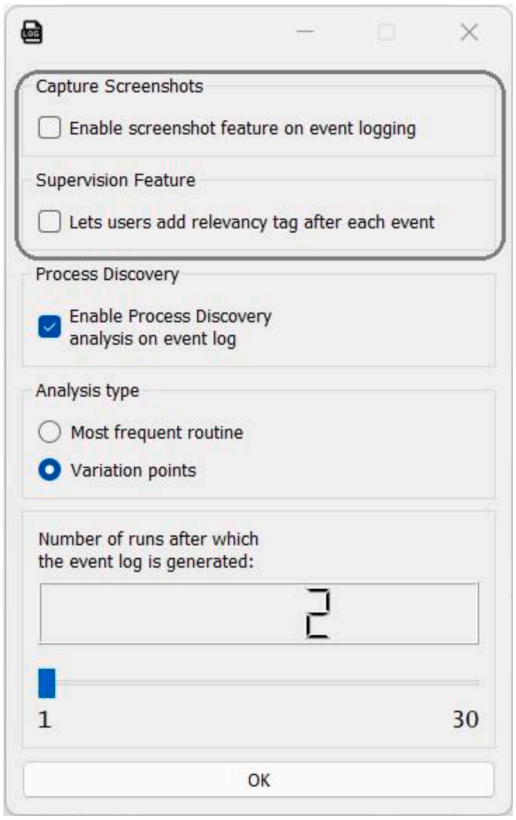


Fig. 5. SmartRPA preferences.

interaction [23]. For the sake of understanding, Fig. 2 provides an excerpt of UI logs produced as output of the Log Processing component. SmartRPA is also able to produce an XES[5] [26] (eXtensible Event Stream) version of the datastream that will contain exactly *n* traces, one for each recorded CSV event log and can be inspected using the most popular process mining tools, such as *ProM*[6] or *Disco*.[7]

The third software component is an **Event Abstraction** engine used to produce a high-level event log from the low-level one with the goal to: *(i)* filter out the noise and irrelevant events for the routine execution. For example, during several training sessions of the routine under analysis, applications related to the operating system may start in the background while the Log Recording component is recording the UI log, and they may dirty the recording phase of the users during their training session. From a workflow perspective, these events are not relevant for any RPA analyst that aims to understand the general behavior of the routine and thus they can be filtered out. While there is a predefined set of irrelevant events, it can be expanded as needed through action tagging, enabled by the supervision feature; *(ii)* group similar low-level events to the same high-level concept. For example, on a web page, the Log Recording component can capture different types of clicks, based on the element clicked. From the RPA analyst perspective, it is not relevant what kind of click was performed, thus the high-level workflow of the routine may show the action "Click on button"; *(iii)* create descriptive labels. Any recorded event provides a low-level description of the UI action performed. To make the UI action underlying an event more descriptive for the RPA analyst, the payload information stored in the low-level event log can be added to its label, such as the cell and the sheet edited, the value inserted, etc. This allows us to create a more descriptive label for any event in the high-level event log, e.g., "*Edit cell B2 on Sheet 'Request' with value 'Full Professor'*".

At this point, the **Process Discovery** component exploits the high-level event log to derive the underlying high-level workflow as a Directly-Follows Graph (DFG), by applying the heuristic miner (the decision to employ the heuristic miner has been driven by its ability to discover highly understandable flowcharts from a BPM analyst perspective [27,28]) implemented in PM4PY [29]. In addition, the knowledge of the workflow underlying the routine, coupled with the low-level version of the dataframe-based event log, will be used to support the identification of different *variation points*, thus leading to the detection of the most suitable routine variant according to intermediate user inputs observed in the low-level dataframe-based event log. A variation point refers to a specific point within a routine execution where a user's choice between different paths must be made.

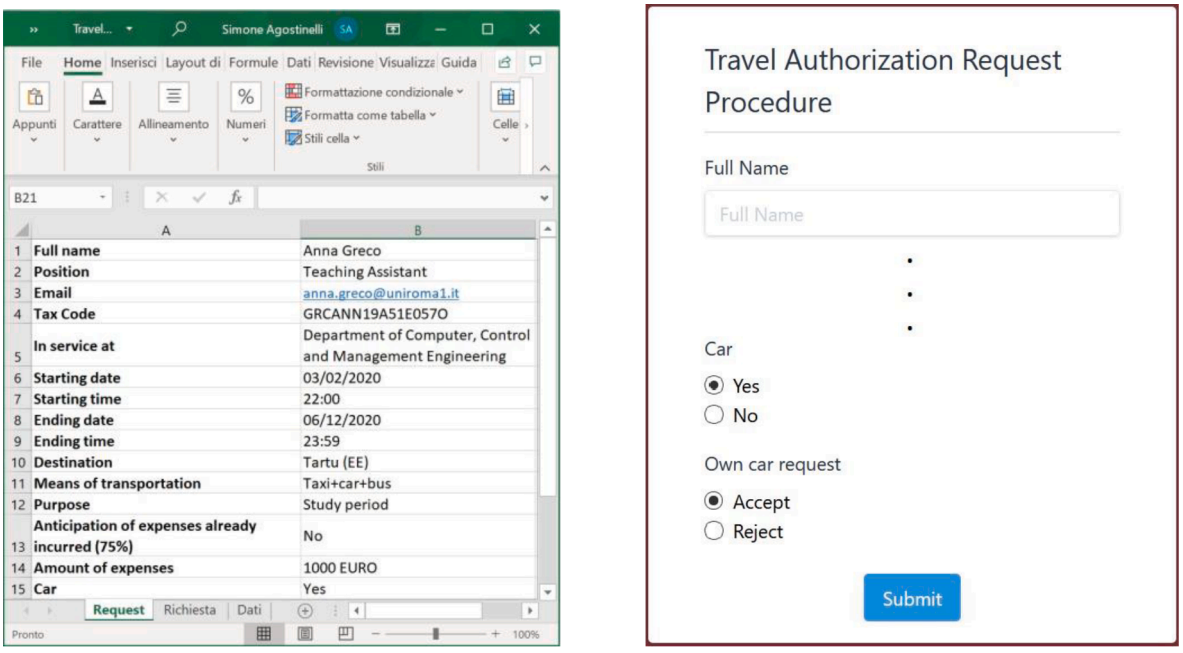
Once the routine variant to automatize is selected, an RPA analyst can personalize the values stored in its events, thanks to the **Script Generation** component. SmartRPA automatically detects the events that can be edited, such as pasting a text or editing an Excel cell, and lets the RPA analyst edit them. After confirmation, the low-level dataframe-based event log is updated. Finally, the Python executable script based on the selected routine variant and updated with the RPA analyst's edits is generated by scanning the recorded low-level events in the dataframe-based log and converting them into executable pieces of software code in Python. The Script Generation component relies on *Automagica*[8] and *Selenium*,[9] a popular suite of tools for process and web browsers automation. Note that the Script Generation component considers only the platform where the software robot is going to be run regardless of the operating system used to record the UI log. For example, if the selected routine variant was recorded on MacOS, but the tool is being executed on Windows, the RPA script will be generated taking into account this aspect, e.g., by converting the information about the system paths. This guarantees cross-platform compatibility across UI logs recorded on different platforms. SmartRPA is also able to generate RPA scripts compatible with the commercial tool *UiPath Studio*.[10]

3. Illustrative example

Below, we introduce a real-life scenario used to demonstrate the major functions of our tool. The example is inspired by the work performed by a University Administration Office, which consists of filling the travel authorization request form made by the personnel of the university for travel requiring prior approval. We specifically consider the task of filling a well-structured Excel spreadsheet (cf. Fig. 6(a)), manually performed by a request applicant that provides some personal information together with further information related to the travel. Then, the spreadsheet is sent via email to an employee of the University Administration Office, which is in charge of processing the request: for each row in the spreadsheet, the employee manually copies every cell in that row and pastes that into the corresponding text field in a dedicated Google form (cf. Fig. 6(b)). In addition, if the request applicant declares the need to use a personal car as one of the means of transport for the travel (by filling the dedicated row labeled with "Car" in the spreadsheet), then the employee has to activate the request on the Google form (in this case, a dialog box labeled "Own car request" appears on the UI, cf. Fig. 6(b)) and then accept or reject the personal car request. Accepting or rejecting the car request represents a *variation point* of the routine, that forks its execution flow into two well-distinguished exclusive branches. When the data transfer for a given travel authorization request has been completed, the employee presses the "Submit" button to confirm data and submit them into an internal database.

The exact steps to correctly perform the routine in Fig. 6 with

[5] XES is the standard for the storage, interchange, and analysis of event logs.
[6] http://www.promtools.org/
[7] https://fluxicon.com/disco/
[8] https://github.com/automagica/automagica
[9] https://www.selenium.dev/
[10] https://www.uipath.com/product/studio



(a) Excel spreadsheet (b) Google form

Fig. 6. UIs involved in the illustrative example.

SmartRPA are the following ones:

1. Tick the checkboxes related to Excel, Clipboard, and the browser installed on the applicant's PC/MAC, and click *"Start logger"*.
2. [Optional] Enable screenshots and supervision features by clicking on the preferences menu as shown in Fig. 5.
3. Open the Excel spreadsheet containing the information about the travel.
4. Open the Google form.
5. Copy and paste each value from the Excel spreadsheet to the Google form.
6. Accept or reject the personal car request (if required).
7. Submit the form.
8. Push the *"Stop logger"* button to stop recording.

At this point, all the operational components explained in Section 2 are executed in sequence to enact the most suitable routine variant according to the detected variation point (either *accepting* or *rejecting* the personal car request) directly from the recorded UI log. A screencast with installation instructions and showing the working of SmartRPA against the illustrative example is available in the GitHub repository of the tool at: https://github.com/bpm-diag/smartRPA/. Additionally, the repository includes a dedicated video showcasing both the screenshot and supervision features.

4. Impact

While RPA is currently used for automating routines and high-volume tasks requiring the manual intervention of expert users, SmartRPA aims to automatically develop software robots directly from the users' observed behavior. SmartRPA offers an innovative contribution to RPA technology to mitigate some of its core downsides. Indeed, differently from the literature approaches to automated RPA script generation from UI logs ([20,21,22,23]), which enable the automation of straightforward routines that have essentially no variance and do not require any human intervention, the software robots generated by SmartRPA are capable of handling the intermediate user inputs that are required during the routine execution, thus enabling emulation of the most suitable routine variant for any specific combination of user inputs as observed in the UI log.

This makes the generation of software robots performed by SmartRPA tailored to any user decision found during a routine execution. This is due to the fact that the behavior of software robots is determined immediately before their enactment, as it is driven by the specific user inputs required to execute the routine. Therefore, SmartRPA acknowledges the benefit of human involvement at multiple points of the routine execution, leveraging the "humans-in-the-loop" model for the automated execution of routines that are less static and require decision-making [12].

In relation to **RQ1**, SmartRPA reduces the dependency on RPA human experts by incorporating mechanisms that automatically generate software robots from observed user behavior, minimizing the need for manual intervention in the software robot creation process. This approach streamlines the RPA lifecycle and enables broader accessibility of RPA technologies. Furthermore, in relation to **RQ2**, the integration of screenshots into the process provides valuable contextual information that enriches the data available for subsequent phases of software robot generation. Screenshots act as a valuable tool to complement raw UI logs, offering additional insights that may be missed when relying solely on purely textual UI logs [19]. From such enriched UI Logs, by using image processing techniques and Machine Learning algorithms, more complete explanations of human routines could be offered. Related to **RQ3**, this inclusion of screenshots not only enriches the dataset but also facilitates the development of a user annotation assistant. This assistant aids in the selective filtering of relevant user actions from the UI logs, enhancing the quality of the input data and consequently improving the automatic generation of software robots. This selective filtering is important to identifying and retaining only the most relevant user actions, thereby refining the software robot's behavior.

Thus, these results can have a relevant impact not just on the RPA community, but also on Human-centered AI communities since SmartRPA promotes collaboration between humans and software robots, aligning with the principles of human-centered AI, which seek to

empower humans and enhance their capabilities rather than replace them. Industries particularly set out to implement RPA with AI leading to Intelligent Process Automation to enhance productivity and enable employees to fulfill more demanding duties [30,31]. SmartRPA streamlines the RPA project lifecycle by automating some of its key steps (cf. steps "(2) model", "(3) record", and "(4) develop"), potentially reducing time and resources required for RPA implementation. This efficiency gain can lead to cost savings and broader adoption of RPA technology across various industries. Additionally, SmartRPA can be integrated into educational programs, providing students with practical experience in RPA project management, automation techniques, and software development. Students can use SmartRPA to gain hands-on experience in automating processes, analyzing UI logs, and generating RPA scripts, thereby enhancing their skill set and readiness for careers in the RPA field.

5. Conclusions

This paper presented SmartRPA, a tool that is able to *(i)* interpret the UI logs enriched with screenshots and annotations that record the mouse/key events that happen on the UI of the software applications involved in many routine executions, *(ii)* discover all the variants (and variation points) of the routine under observation, and *(iii)* automatically combine them into an executable RPA script, which can be enacted by means of a single software robot. SmartRPA extends the RPM pipeline with five different stages, as described in Section 2.

Notably, using SmartRPA, all the routine executions recorded by the tool can be automated, a high-level flowchart diagram is presented to expert users for potential diagnosis operations, and screenshots of recorded user actions can be inspected to improve understanding of the underlying process behavior. The executable RPA script to drive the working of a software robot is generated by solely interpreting the routine executions stored in the UI log, selecting step-by-step the most suitable routine variant. Also, the tool enables customizing input fields within the selected routine variant prior to the execution of the related RPA script, thus supporting those steps that require intermediate manual user inputs. As a consequence, this makes the working of software robots flexible and adaptable to several situations, thanks also to the annotation feature, which enables users to tag actions as relevant or irrelevant to the process being automated.

The main weakness of SmartRPA is correlated with the quality of information recorded in real-world UI logs. Since a UI log is fine-grained, routines executed with many different strategies may potentially affect the identification of the variation points. For this reason, the current version of the software tool can be improved in the future to emulate routines executed with many different strategies.

Moreover, in SmartRPA we have currently neglected the segmentation issue [32]. Segmentation is the challenge to automatically understand which user actions contribute to which routines inside a UI log, which is trivially already solved in SmartRPA because UI logs are generated through controlled training sessions. However, future developments should incorporate a segmentation component into the SmartRPA architecture, facilitating the analysis of unsegmented UI logs acquired from more inclusive training sessions that are not focused just on single routines [33–35].

CRediT authorship contribution statement

S. Agostinelli: Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Data curation. **T. Hohenadl:** Visualization, Software, Resources, Data curation. **A. Marrella:** Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization. **A. Martínez-Rojas:** Software, Data curation, Resources, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The work of S. Agostinelli has been supported by the PNRR MUR project PE0000013-FAIR. The work of A. Marrella is supported by the H2020 project DataCloud (grant number 101016835) and the Sapienza project FOND-AIBPM (Foundations of AI-Augmented Business Process Management). The work of A. Martínez-Rojas has been supported by the EQUAVEL project PID2022-137646OB-C31 funded by MICIU/AEI/10.13039/501100011033 and by "ERDF/EU"; the FPU scholarship (FPU20/05984) funded by MICIU/AEI/10.13039/501100011033 and, by "ESF+", and its mobility grant (EST24/00631).

References

[1] Ramirez AJ, Reijers HA, Barba I, Valle CD. A method to improve the early stages of the robotic process automation lifecycle. In: Giorgini P, Weber B, editors. 31st Int. Conf. on Advanced Information Systems Engineering (CAiSE'19). Springer; 2019. p. 446–61. https://doi.org/10.1007/978-3-030-21290-2_28. Vol. 11483.
[2] Herm L-V, Janiesch C, Helm A, Imgrund F, Fuchs K, Hofmann A, Winkelmann A. A consolidated framework for implementing robotic process automation projects. In: 18th Int. Conf. on Business Process Management, BPM'20. Springer; 2020. p. 471–88. https://doi.org/10.1007/978-3-030-58666-9_27.
[3] van der Aalst WMP. Robotic process automation. Bus Inf Syst Eng 2018;60(4). https://doi.org/10.1007/S12599-018-0542-4.
[4] Penttinen E, Kasslin H, Asatiani A. How to choose between robotic process automation and back-end system automation?. In: European Conference on Information Systems (ECIS'18); 2018. https://aisel.aisnet.org/ecis2018_rp/66.
[5] R. Ravn, P. Halberg, J. Gustafsson, J. Groes, Get Ready for Robots: why Planning Makes the Difference Between Success and Disappointment, https://eyfinancialservicesthoughtgallery.ie/wp-content/uploads/2016/11/ey-get-ready-for-robots.pdf, Accessed: 19-07-2021 (2016).
[6] Langmann C, Turi D, et al. Robotic process automation (RPA)Digitalisierung und automatisierung von Prozessen. Springer Books; 2020. https://doi.org/10.1007/978-3-658-28299-8.
[7] Agostinelli S, Lupia M, Marrella A, Mecella M. Smartrpa: a tool to reactively synthesize software robots from user interface logs. In: 33rd Int. Conf. on Adv. Inf. Sys. Eng. Forum (CAiSE'21). Springer; 2021. p. 137–45. https://doi.org/10.1007/978-3-030-79108-7_16. Vol. 424.
[8] Agostinelli S, Lupia M, Marrella A, Mecella M. Automated generation of executable RPA scripts from user interface logs. In: 18th Int. Conf. on Business Process Management (BPM'20) - Blockchain and Robotic Process Automation. Springer; 2020. p. 116–31. https://doi.org/10.1007/978-3-030-58779-6_8. Vol. 393.
[9] Agostinelli S, Lupia M, Marrella A, Mecella M. Reactive synthesis of software robots in RPA from user interface logs. Comput Ind 2022;142:103721. https://doi.org/10.1016/J.COMPIND.2022.103721.
[10] Agostinelli S, Marrella A, Mecella M. Research challenges for intelligent robotic process automation. In: 17th Int. Conf. on Business Process Management Workshops BPM'19. 362. Springer; 2019. p. 12–8. https://doi.org/10.1007/978-3-030-37453-2_2.
[11] S. Agostinelli, A. Marrella, M. Mecella, Towards Intelligent Robotic Process Automation for BPMers, in: AAAI-20 Workshop on Intelligent Process Automation, Vol. abs/2001.00804, 2020. arXiv:2001.00804.
[12] Chakraborti T, Isahagian V, Khalaf R, Khazaeni Y, Muthusamy V, Rizk Y, Unuvar M. From robotic process automation to intelligent process automation - emerging trends. In: 18th Int. Conf. on Business Process Management (BPM'20) - Blockchain and Robotic Process Automation Forum. Springer; 2020. p. 215–28. https://doi.org/10.1007/978-3-030-58779-6_15. Vol. 393.
[13] Leno V, Polyvyanyy A, Dumas M, Rosa ML, Maggi FM. Robotic process mining: vision and challenges. Bus Inf Syst Eng 2021;63(3):301–14. https://doi.org/10.1007/S12599-020-00641-4.
[14] Rehse J-R, Abb L, Berg G, Bormann C, Kampik T, Warmuth C. User behavior mining: a research agenda. Business and information systems engineering. 2024. https://doi.org/10.1007/s12599-023-00848-1.
[15] Leno V, Polyvyanyy A, Rosa ML, Dumas M, Maggi FM. Action logger: enabling process mining for robotic process automation. In: Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at the 17th Int. Conf. on Business Process Management (BPM'19); 2019. p. 124–8. Vol. 2420, CEUR-WS.org, https://ceur-ws.org/Vol-2420/paperDT2.pdf.
[16] Bosco A, Augusto A, Dumas M, La Rosa M, Fortino G. Discovering automatable routines from user interaction logs. In: 17th Int. Conf. on Business Process Management Forum (BPM'19). Springer; 2019. p. 144–62. https://doi.org/10.1007/978-3-030-26643-1_9.
[17] Linn C, et al. Desktop Activity Mining - A New Level of Detail in Mining Business Processes. INFORMATIK 2018. 2018. p. 245–58.

[18] Martínez-Rojas A, Ramirez AJ, Enríquez JG, Reijers HA. Analyzing variable human actions for robotic process automation. In: 20th Int. Conf. on Business Process Management (BPM'22). Springer; 2022. p. 75–90. https://doi.org/10.1007/978-3-031-16103-2_8. Vol. 13420.

[19] Martínez-Rojas A, Ramirez AJ, Enríquez JG, Reijers HA. A screenshot-based task mining framework for disclosing the drivers behind variable human actions. Inf Syst 2023;121:102340. https://doi.org/10.1016/j.is.2023.102340.

[20] Gao J, van Zelst SJ, Lu X, van der Aalst WMP. Automated robotic process automation: a self-learning approach. In: On the Move to Meaningful Internet Systems: OTM 2019 Conf. Springer; 2019. p. 95–112. https://doi.org/10.1007/978-3-030-33246-4_6.

[21] Le V, Gulwani S. FlashExtract: a Framework for Data Extraction by Examples. ACM SIGPLAN PLDI '14. 2014. p. 542–53. https://doi.org/10.1145/2594291.2594333.

[22] Miltner A, Gulwani S, Le V, Leung A, Radhakrishna A, Soares G, Tiwari A, Udupa A. On the fly synthesis of edit suggestions. ACM Program Lang 2019;3. https://doi.org/10.1145/3360569 (OOPSLA)143:1–143:29.

[23] Leno V, Deviatykh S, Polyvyanyy A, Rosa ML, Dumas M, Maggi FM. Robidium: automated synthesis of robotic process automation scripts from UI logs. In: Demonstration & Resources Track at 18th Int. Conf. on Business Process Management (BPM'20); 2020. p. 102–6. http://ceur-ws.org/Vol-2673/paperDR08.pdf.

[24] Beyel HH, van der Aalst WMP. Creating translucent event logs to improve process discovery. In: Int. Conf. on Process Mining (ICPM'22). Springer; 2022. p. 435–47. https://doi.org/10.1007/978-3-031-27815-0_32.

[25] Meironke A, Kuehnel S. How to measure RPA's benefits? A review on metrics, indicators, and evaluation methods of RPA benefit assessment. In: 17th Int. Tagung Wirtschaftsinformatik (WI'22). AISeL; 2022. https://aisel.aisnet.org/wi2022/bpm/bpm/5.

[26] IEEE standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. IEEE Std 1849-2023 (Revision of IEEE Std 1849-2016) 2023:1–55. https://doi.org/10.1109/IEEESTD.2023.10267858.

[27] Agostinelli S, Maggi FM, Marrella A, Milani F. A user evaluation of process discovery algorithms in a software engineering company. In: IEEE 23rd Int. Enterprise Distributed Object Computing Conference (EDOC'19); 2019. p. 142–50. https://doi.org/10.1109/EDOC.2019.00026.

[28] Agostinelli S, Covino F, D'Agnese G, De Crea C, Leotta F, Marrella A. Supporting governance in healthcare through process mining: a case study. IEEE Access 2020;8:186012–25.

[29] A. Berti, S.J. van Zelst, W. van der Aalst, Process Mining for Python (PM4Py): bridging the Gap Between Process- and Data Science (2019). arXiv:1905.06169.

[30] Siderska J, Aunimo L, Süße T, von Stamm J, Kedziora D, Aini SNBM. Towards Intelligent Automation (IA): literature review on the evolution of Robotic Process Automation (RPA), its challenges, and future trends. Eng Manag in Prod and Serv 2023;15(4):90–103.

[31] Kedziora D, Siemon D, Sharda R. 'Leanbotics' Case - Exploring Inclusive Synergies Between Robotic Process Automation (RPA) and process improvement. In: 57th Hawaii International Conference on System Sciences, HICSS 2024; 2024. p. 1628–37. ScholarSpace, https://hdl.handle.net/10125/106582.

[32] Agostinelli S, Marrella A, Mecella M. Automated segmentation of user interface logs. Robotic process automation management: technology, applications, de gruyter oldenbourg. 2021. p. 201–22. https://doi.org/10.1515/9783110676693-011.

[33] Leno V, Augusto A, Dumas M, La Rosa M, Maggi FM, Polyvyanyy A. Identifying candidate routines for robotic process automation from unsegmented UI logs. In: 2nd Int. Conf. on Process Mining (ICPM'20); 2020. p. 153–60. https://doi.org/10.1109/ICPM49681.2020.00031.

[34] Agostinelli S, Leotta F, Marrella A. Interactive segmentation of user interface logs. In: 19th Int. Conf. on Service-Oriented Computing (ICSOC'21); 2021. p. 65–80. https://doi.org/10.1007/978-3-030-91431-8_5. Vol. 13121.

[35] Rebmann A, van der Aa H. Unsupervised task recognition from user interaction streams. In: 35th Int. Conf. on Advanced Information Systems Engineering (CAiSE'23). Springer; 2023. p. 141–57. https://doi.org/10.1007/978-3-031-34560-9_9. Vol. 13901.